

LAPORAN PRAKTIKUM STRUKTUR DATA
LINKED LIST PADA JAVA



DOSEN PENGAMPU :

Dr. Wahyudi, M.T.

OLEH :

RIFKI YULIANDRA

NIM.2311532011

DEPARTEMEN INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS ANDALAS
PADANG

2024

Linked List Pada Java

I. TUJUAN

- 1.1. Memahami apa yang dimaksud dengan Linked List
- 1.2. Memahami cara membuat Single Linked List sederhana dengan class node terpisah
- 1.3. Memahami cara membuat Single Linked List dengan mengimport dari library Java
- 1.4. Memahami cara membuat Double Linked List

II. TEORI

Linked list merupakan struktur data linier yang tersusun dari elemen-elemen yang disebut node. Setiap node memiliki dua komponen penting: data dan pointer. Data menyimpan nilai yang ingin kita simpan, sedangkan pointer menunjuk ke node berikutnya dalam daftar. Tidak seperti array yang memiliki urutan memori yang berurutan, linked list dihubungkan melalui pointer.

Terdapat beberapa jenis linked list, di antaranya singly linked list, doubly linked list, dan circular linked list. Masing-masing jenis memiliki perbedaan pada arah dan jumlah pointer yang dimiliki setiap node.

Operasi dasar pada linked list meliputi penyisipan, penghapusan, pencarian, dan traversal. Penyisipan dan penghapusan node dapat dilakukan dengan mudah tanpa menggeser elemen lain, menjadikannya struktur data yang fleksibel. Linked list juga terbilang efisien dalam penggunaan memori karena hanya membutuhkan memori untuk node yang terisi, tidak seperti array yang harus mengalokasikan ruang untuk semua elemen, meskipun beberapa kosong.

Meskipun memiliki kelebihan, linked list juga memiliki beberapa kekurangan. Salah satu kekurangannya adalah akses acak yang lambat. Untuk mengakses elemen tertentu dalam daftar, kita perlu melewati setiap node dari awal hingga menemukan node yang diinginkan, membutuhkan waktu $O(n)$, di mana n adalah jumlah node dalam daftar.

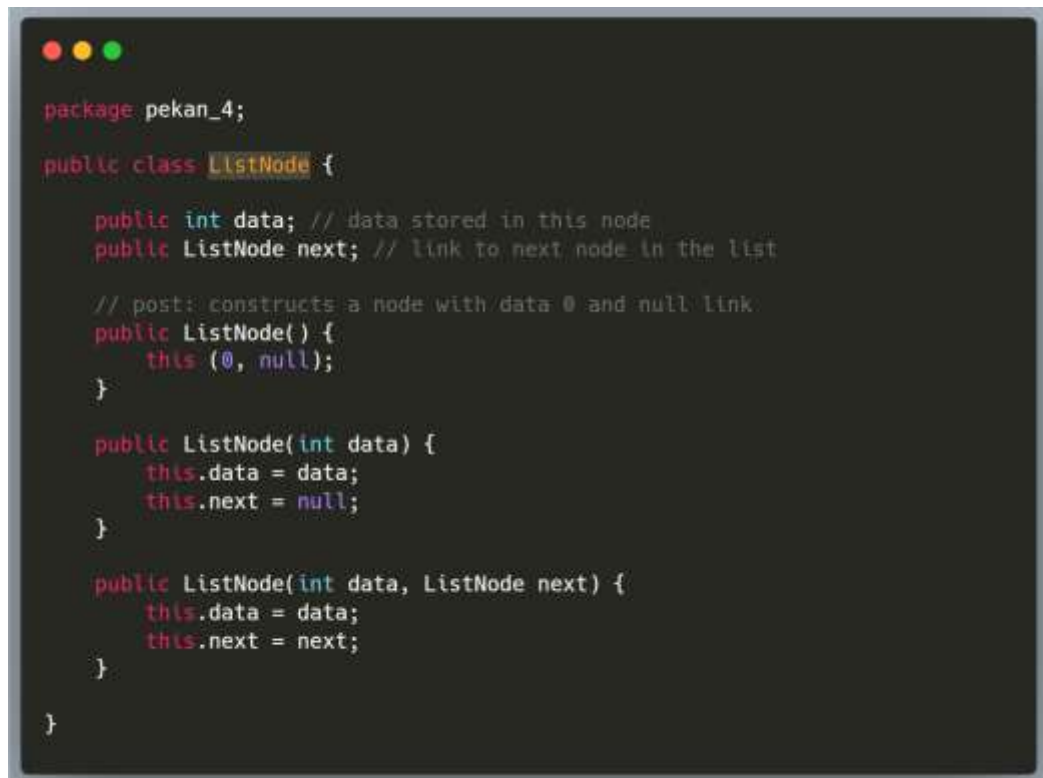
Kekurangan lainnya adalah penggunaan memori overhead. Setiap node membutuhkan memori tambahan untuk menyimpan pointer. Selain itu, implementasi

algoritma tertentu, seperti pengurutan, mungkin lebih kompleks pada linked list dibandingkan dengan struktur data lain.

Dalam praktikum ini akan dilakukan proses program linked list seperti Single Linked List dan juga Double Linked List.

III. LANGKAH KERJA PRAKTIKUM

3.1. Single Linked List 1



```
package pekan_4;

public class ListNode {

    public int data; // data stored in this node
    public ListNode next; // link to next node in the list

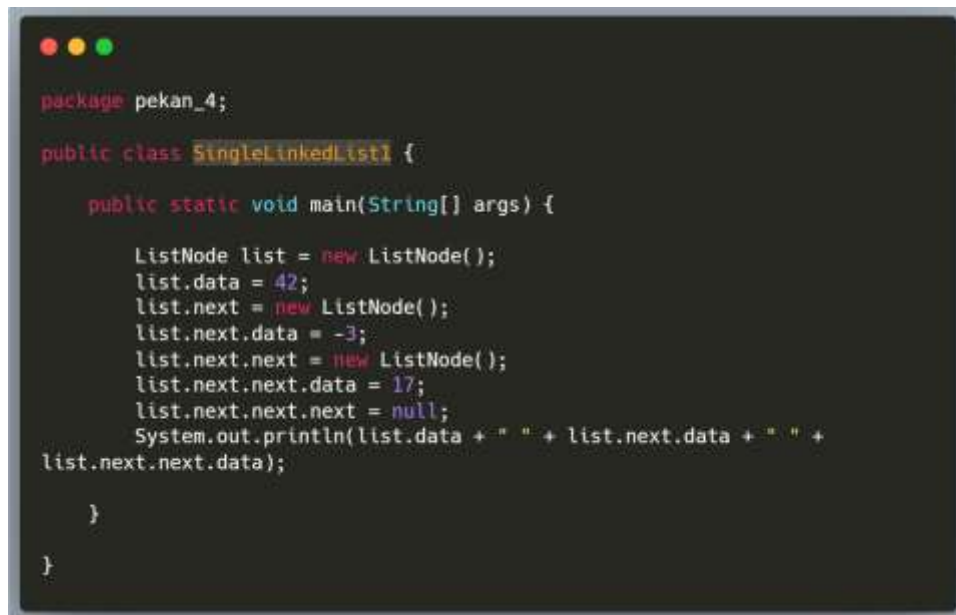
    // post: constructs a node with data 0 and null link
    public ListNode() {
        this(0, null);
    }

    public ListNode(int data) {
        this.data = data;
        this.next = null;
    }

    public ListNode(int data, ListNode next) {
        this.data = data;
        this.next = next;
    }

}
```

Pertama kita akan membuat sebuah class untuk membuat node-node dari sebuah linked list. Setiap node dalam linked list terdiri dari dua bagian, yaitu data dan link. Data disimpan dalam atribut 'data', sedangkan link ke node berikutnya disimpan dalam atribut 'next'.



```

package pekan_4;

public class SingleLinkedList1 {

    public static void main(String[] args) {

        ListNode list = new ListNode();
        list.data = 42;
        list.next = new ListNode();
        list.next.data = -3;
        list.next.next = new ListNode();
        list.next.next.data = 17;
        list.next.next.next = null;
        System.out.println(list.data + " " + list.next.data + " " +
list.next.next.data);

    }

}

```

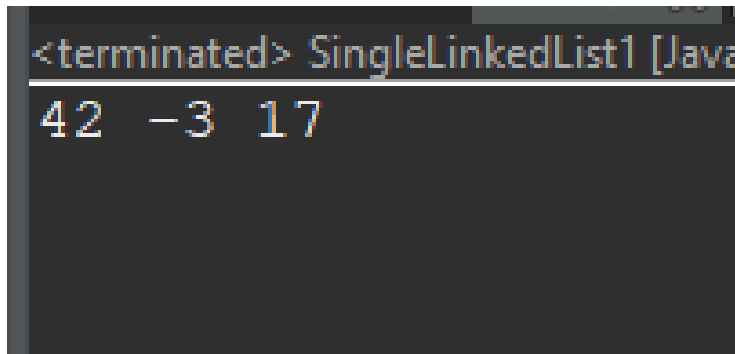
Selanjutnya kita membuat sebuah linked list sederhana seperti gambar program diatas. Pada baris pertama, kita membuat sebuah objek 'ListNode' bernama 'list' dengan memanggil konstruktor kosong. Setelah itu, kita menetapkan nilai dari atribut 'data' pada node pertama dengan mengisi nilainya dengan 42.

Pada baris ketiga, kita membuat sebuah node baru dengan memanggil konstruktor 'ListNode()' tanpa parameter. Setelah itu, kita menetapkan node baru sebagai link dari node pertama dengan mengisi atribut 'next' dengan node baru tersebut.

Pada baris keempat sampai keenam, kita melakukan hal yang sama seperti pada baris kedua dan ketiga, tetapi untuk node kedua dan ketiga. Kita menetapkan nilai dari atribut 'data' pada setiap node dengan mengisi nilainya dengan -3 dan 17. Setelah itu, kita menetapkan link dari setiap node ke node berikutnya dengan mengisi atribut 'next' dari setiap node.

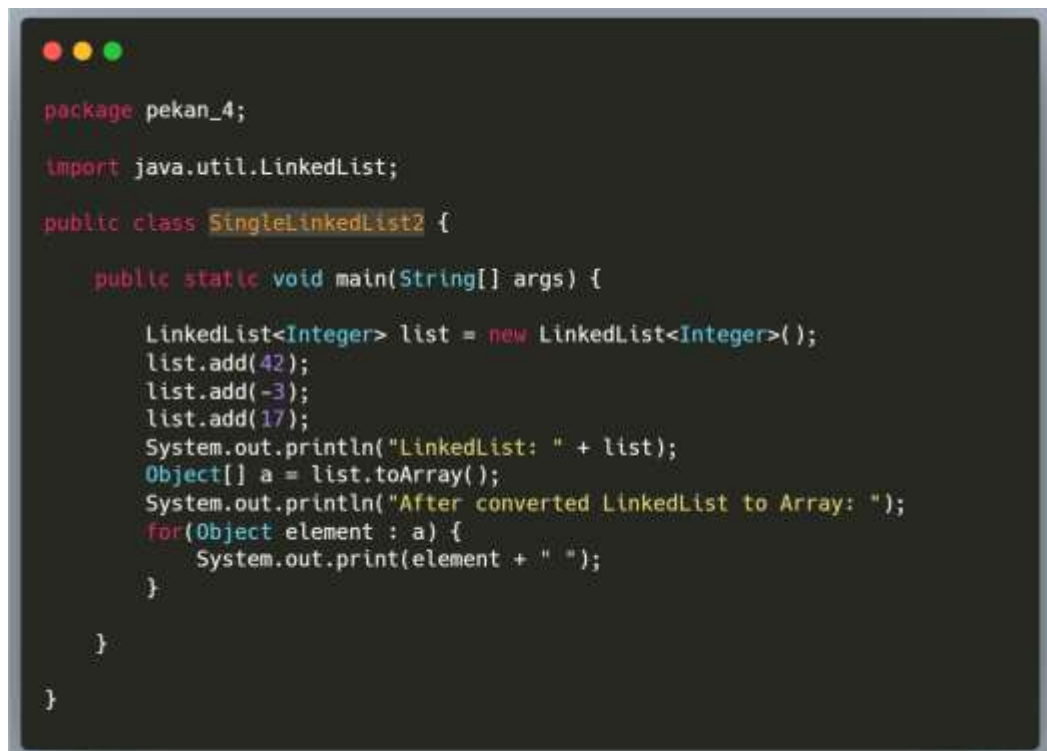
Pada baris terakhir, kita mencetak nilai dari atribut 'data' pada setiap node dengan menggunakan perintah 'System.out.println()'. Hasil dari cetakan akan menampilkan tiga nilai yang disimpan pada setiap node, yaitu 42, -3, dan 17.

Adapun output yang dihasilkan :



```
<terminated> SingleLinkedList1 [Java
42 -3 17
```

3.2. Single Linked List 2



```
package pekan_4;

import java.util.LinkedList;

public class SingleLinkedList2 {

    public static void main(String[] args) {

        LinkedList<Integer> list = new LinkedList<Integer>();
        list.add(42);
        list.add(-3);
        list.add(17);
        System.out.println("LinkedList: " + list);
        Object[] a = list.toArray();
        System.out.println("After converted LinkedList to Array: ");
        for(Object element : a) {
            System.out.print(element + " ");
        }

    }

}
```

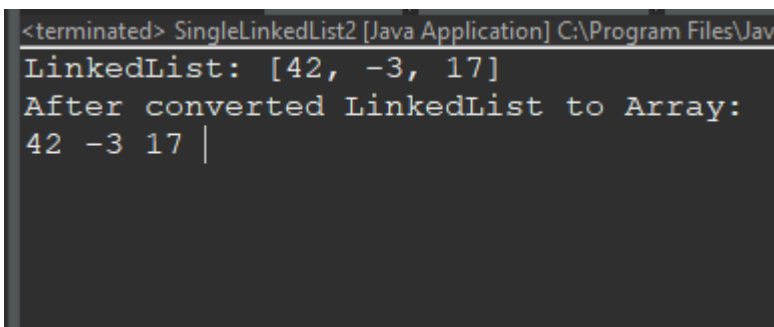
Pada baris pertama, kita import kelas 'LinkedList' dari package 'java.util'. Pada baris ketiga, kita membuat sebuah objek 'LinkedList' bernama **list** yang dapat menyimpan nilai-nilai integer. Pada baris keempat sampai keenam, kita menambahkan nilai-nilai 42, -3, dan 17 ke dalam linked list menggunakan metode 'add()'.

Pada baris ketujuh, kita mencetak isi dari linked list menggunakan perintah 'System.out.println()'. Hasil dari cetakan akan menampilkan isi dari linked list dalam bentuk string, seperti [42, -3, 17].

Pada baris kedelapan, kita mengkonversi linked list menjadi sebuah array menggunakan metode 'toArray()'. Metode ini akan mengembalikan sebuah array yang berisi semua elemen dari linked list.

Pada baris kesembilan, kita mencetak isi dari array yang telah dikonversi menggunakan perintah 'System.out.println()'. Kita menggunakan loop **for-each** untuk mencetak setiap elemen dari array.

Adapun Output yang dihasilkan :



```
<terminated> SingleLinkedList2 [Java Application] C:\Program Files\Java\bin\java.exe
LinkedList: [42, -3, 17]
After converted LinkedList to Array:
42 -3 17 |
```

Perbedaan antara program ini dengan program sebelumnya adalah bahwa program ini menggunakan kelas 'LinkedList' dari Java, sedangkan program sebelumnya menggunakan kelas 'ListNode' yang dibuat sendiri. Kelas 'LinkedList' dari Java telah menyediakan metode-metode yang berguna untuk mengelola linked list, seperti 'add()', 'remove()', dan 'toArray()', sehingga kita tidak perlu membuat sendiri.

3.3. Double Linked List

```
package pekan_4;

public class DoubleLinkedList3 {
    static Node head;
    class Node{
        int data;
        Node prev;
        Node next;
        Node(int d){ data = d;}
    }

    public void push(int new_data) {
        Node new_Node = new Node(new_data);
        new_Node.next = head;
        new_Node.prev = null;
        if(head != null)
            head.prev = new_Node;
        head = new_Node;
    }

    public void display(Node head) {
        Node temp = head;
        while(temp != null) {
            System.out.print(temp.data + " --> ");
            temp = temp.next;
        }
        System.out.println("Null");
    }

    public static void main(String[] args) {
        DoubleLinkedList3 dll = new DoubleLinkedList3();
        dll.push(42);
        dll.push(-3);
        dll.push(17);
        dll.display(head);
    }
}
```

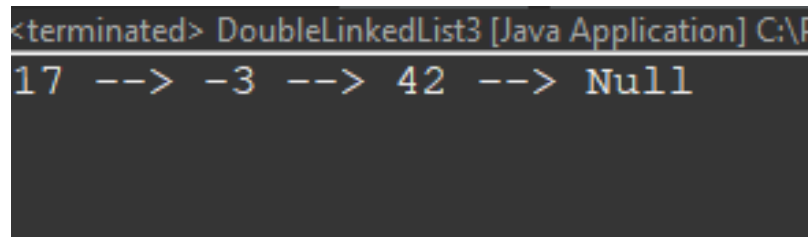
Pada baris ketiga hingga delapan, kita membuat kelas 'Node' yang digunakan untuk menyimpan data dan referensi ke node sebelumnya dan node berikutnya.

Pada baris sepuluh hingga sediri belas, kita membuat metode 'push()' yang digunakan untuk menambahkan node baru ke awal linked list. Metode ini membuat node baru, mengatur referensi ke node sebelumnya dan node berikutnya, dan mengubah referensi ke awal linked list jika linked list tidak kosong.

Pada baris dua belas hingga tiga puluh, kita membuat metode 'display()' yang digunakan untuk mencetak isi linked list. Metode ini memulai dari node awal dan mencetak data dari setiap node sampai node akhir.

Pada baris tiga puluh satu hingga tiga puluh tiga, kita membuat kelas utama 'DoubleLinkedList3' dan membuat objek 'dll' dari kelas tersebut. Kemudian, kita menambahkan beberapa data ke linked list menggunakan metode 'push()'. Selanjutnya, kita mencetak isi linked list menggunakan metode 'display()'.

Adapun output yang dihasilkan ketika dijalankan:



```
<terminated> DoubleLinkedList3 [Java Application] C:\P
17 --> -3 --> 42 --> Null
```

IV. KESIMPULAN

Linked list adalah struktur data yang digunakan untuk menyimpan data dalam bentuk node yang saling terhubung. Linked list dapat dibuat dalam bentuk satu arah atau dua arah, dan dapat menggunakan kelas 'LinkedList' dari Java atau kelas 'ListNode' yang dibuat sendiri.

Dalam program pertama, kita membuat linked list satu arah (Single Linked List) menggunakan kelas 'ListNode' yang dibuat sendiri. Dalam program kedua, kita menggunakan kelas 'LinkedList' dari Java untuk membuat linked list satu arah (Single Linked List) dan mengkonversinya menjadi sebuah array. Dalam program ketiga, kita membuat linked list dua arah menggunakan kelas Node yang dibuat sendiri. Dalam program keempat, kita menggunakan kelas 'DoubleLinkedList3' yang mengimplementasikan linked list dua arah (Double Linked List). Dengan menggunakan linked list, kita dapat melakukan operasi seperti penambahan, penghapusan, dan pencarian data dengan efisien.